Génération aléatoire de séquences génomiques structurées

Alain Denise*	Olivier Roques	Michel Termier
LRI† & IGM‡	LaBRI§	IGM
Univ. Paris-Sud XI	Univ. Bordeaux I	Univ. Paris-Sud XI

Résumé

Nous présentons une méthodologie de génération de séquences aléatoires soumises à la fois à des contraintes statistiques et structurelles. Pour ce faire, nous modélisons les séquences par des langages formels non contextuels (ou algébriques) et imposons une loi de probabilité sur les nombres d'occurrences des lettres.

1 Introduction

Les séquences aléatoires sont couramment utilisées pour aider à la découverte de caractéristiques nouvelles de séquences biologiques. La démarche consiste à comparer les propriétés de séquences aléatoires obéissant à un modèle donné avec des propriétés observées sur des séquences réelles. Des différences observées, on déduit des faits biologiques. En quelque sorte, les séquences aléatoires représentent le «bruit de fond» à partir duquel, par contraste, on cherche à déterminer quels sont les éléments biologiquement pertinents dans les séquences réelles. Un exemple emblématique d'utilisation des séquences aléatoires est la recherche de motifs exceptionnels pour identifier des sites d'intérêt biologique. L'hypothèse de départ –souvent avérée– est qu'un motif exceptionnellement sur-représenté dans une séquence biologique est le signe d'une fonctionnalité biologique. Inversement, une sous-représentation peut indiquer un motif évité car néfaste pour l'organisme. La sur- ou sous-représentation est mesurée en comparant le nombre d'occurrences observées dans les séquences naturelles et celui attendu dans des séquences aléatoires obéissant à un modèle exploratoire donné [20]. Le problème éminemment important de l'identification des zones promotrices fait appel à ce type de méthode : on recherche des motifs souvent présents dans des zones connues pour contenir des promoteurs [27, 32]. D'autres champs de l'analyse des séquences font appel de manière similaire à des modèles exploratoires. Il en est ainsi de la recherche d'homologies par alignement : pour évaluer la pertinence biologique des scores d'alignement obtenus, on les compare à ceux obtenus sur des séquences aléatoires [1, 16, 17]). Les séquences aléatoires sont aussi employées pour tester ou étalonner des algorithmes et logiciels d'analyse et de traitement des séquences [2, 18]). Elles peuvent être partie intégrante d'algorithmes de traitement de séquences. C'est le cas dans [15], où la détection de répétitions en tandem fait appel à l'évaluation de la sur-représentation de motifs.

La grande majorité des modèles de séquences aléatoires utilisés sont fondés sur des critères purement statistiques mesurés sur les séquences réelles. Les bases de ces modèles et les premiers

*Alain.Denise@lri.fr

†LRI, UMR CNRS 8623, bât. 490, Université Paris-Sud XI, 91405 Orsay cedex.

‡IGM, UMR CNRS 8621, bât. 400, Université Paris-Sud XI, 91405 Orsay cedex.

§LaBRI, UMR CNRS 5800, Université Bordeaux I, 33405 Talence.

algorithmes de génération ont été décrits dans [9]. Les paramètres pris en compte sont les fréquences observées des nucléotides ou oligonucléotides de longueur fixée l dans la séquence naturelle de référence. Ainsi par exemple, pour $l = 2$ et la séquence de base *aatgtaacgt*, les fréquences observées sont $aa = 2$, $at = 1$, $tg = 1$, $gt = 2$, $ta = 1$, $ac = 1$ et $cg = 1$. Selon le modèle retenu, on engendre des séquences aléatoires qui respectent les fréquences ainsi mesurées, soit exactement, soit en moyenne. Dans ce dernier cas, la génération s'effectue selon une chaîne de Markov d'ordre $l-1$: à tout moment, la probabilité d'engendrer une lettre donnée dépend des $l-1$ lettres précédemment engendrées. (Si $l = 1$, le lettres sont engendrées indépendamment selon une loi de Bernoulli.) Le processus est clairement linéaire en fonction de la longueur de la séquence engendrée. En fait, dans ce modèle, on sait dans certains cas caractériser analytiquement les lois des fréquences d'apparition de motifs donnés dans des séquences aléatoires [21, 24, 25, 26, 29]. Cela permet donc d'une part de s'affranchir de la génération d'un nombre important de séquences, d'autre part d'obtenir des valeurs exactes des paramètres recherchés. Toutefois, la génération aléatoire reste indispensable dès lors que les propriétés étudiées ne concernent pas des motifs simples (rationnels par exemple). La génération aléatoire en fréquences exactes est un problème plus difficile si l'on désire que le tirage soit uniforme parmi toutes les séquences possibles. Ce problème est résolu dans [14], en profitant du fait, signalé dans [9], qu'il revient à engendrer un chemin eulérien aléatoire dans un graphe ; une implémentation de l'algorithme est présentée dans [3].

Dans ce contexte, notre travail vise à engendrer des séquences aléatoires plus «réalistes» car tenant compte à la fois de critères statistiques et syntaxiques. Il s'agit de tirer au hasard des mots d'un langage algébrique en respectant une distribution donnée sur les nombres d'occurrences de chacune des lettres de l'alphabet. Le problème de la génération aléatoire uniforme (c'est-à-dire équiprobable) de mots de langages a été souvent traité. En revanche celui de la génération aléatoire en imposant une distribution différente de l'uniformité ne l'a pas été jusqu'ici. Le problème est le suivant. On se donne un langage L sur un alphabet $X = \{x_1, x_2, \ldots, x_k\}$ et un entier n. Notons L_n l'ensemble desd mots de L de longueur n. Il s'agit d'engendrer des mots de L_n de façon à respecter une distribution des lettres donnée par un vecteur de k nombres positifs. Nous considérons deux variantes :

1. *Génération en fréquences exactes.* La distribution du nombre de lettres de tout mot engendré doit respecter exactement un vecteur d'entiers $(n_1, \ldots, n_k)$ donné. Il s'agit d'un tirage aléatoire uniforme dans un sous ensemble de L_n constitué de tous les mots $w \in L_n$ tels que $|w|_{x_i} = n_i$ pour tout $i \in \{1, 2, \ldots, k\}$.

2. *Génération en fréquences moyennes.* Les mots engendrés doivent respecter *en moyenne* une distribution donnée par un vecteur $\mathbf{v} = (v_1, \ldots, v_k)$ tel que $v_1 + \ldots + v_k = 1$. Plus précisément, il s'agit de tirer aléatoirement des mots de L_n de façon que

   (a) tout mot de L_n ait une probabilité non nulle d'être engendré ;

   (b) pour tout $i \in \{1, 2, \ldots, k\}$, la proportion de lettres x_i d'un mot engendré soit égale (ou asymptotiquement équivalente) à v_i : si $p(w)$ est la probabilité que le mot w soit engendré par l'algorithme, on doit avoir $\frac{1}{n} \sum_{w \in L_n} |w|_{x_i} p(w) \sim v_i$;

   (c) deux mots ayant la même distribution des lettres aient la même probabilité d'être engendrés.

Notre approche se base sur la méthode dite *récursive* de génération aléatoire uniforme, initiée par Wilf et Nijenhuis [22] puis généralisée et formalisée par Flajolet, Zimmermann et Van Cutsem [10]. La section 2 est consacrée à une brève présentation de cette méthodologie dans le cadre des langages algébriques. Nous présentons dans la section 3 une simple adaptation qui permet d'engendrer des mots en fréquences exactes. Dans la section 4, nous nous intéressons à la génération en fréquences moyennes. Enfin, la section 5 décrit quelques exemples d'application à des séquences génomiques.

2 Génération uniforme

La méthodologie générale de la génération aléatoire uniforme des structures décomposables (qui comprennent les langages algébriques) a été présentée en détail dans [10]. Fondés sur cette méthodologie, des algorithmes particulièrement adaptés au cas particulier des langages algébriques ont été présentés et étudiés dans [19] et [11]. Nous présentons ici la version de la méthode que nous estimons la plus simple, en engageant le lecteur à se reporter aux articles cités pour des variantes plus performantes.

Le point de départ est une grammaire algébrique non ambiguë en forme normale de Chomsky : tout membre droit d'une règle est soit le mot vide ε, soit une lettre de X, soit un symbole non terminal, soit un produit (concaténation) de deux symboles non terminaux. On impose de plus qu'un symbole non terminal ne peut être membre gauche que de deux règles au maximum ; et dans ce cas chacun des membres droits est constitué d'un unique symbole non terminal.

La première étape est une étape de dénombrement : il s'agit de calculer, pour tout symbole non terminal T et pour tout $0 \leq j \leq n$, le nombre $T(i)$ de mots de longueur j qui dérivent de T (rappelons que n est la longueur des mots à engendrer). Ceci s'effectue grâce à des relations de récurrences qui se déduisent directement des règles de la grammaire :

$$T \to \varepsilon \quad \Rightarrow \quad T(0) = 1\ ; \tag{1}$$

$$T \to x_i \quad \Rightarrow \quad T(1) = 1\ ; \tag{2}$$

$$T \to T' \mid T'' \quad \Rightarrow \quad T(n) = T'(n) + T''(n)\ ; \tag{3}$$

$$T \to T'T'' \quad \Rightarrow \quad T(n) = \sum_{n'+n''=n} T'(n')T''(n''). \tag{4}$$

Notons que, du fait que la série génératrice d'un langage algébrique est holonome, ces mêmes coefficients peuvent être calculés avec des récurrences linéaires, ce qui accélère significativement le traitement (voir par exemple [11]).

Cette phase de prétraitement n'est effectuée qu'une fois, quel que soit le nombre de mots de taille n (ou inférieure) que l'on désire engendrer. On obtient un mot aléatoire en effectuant une suite de dérivations à partir de l'axiome de la grammaire en choisissant, à chaque étape, une dérivation avec la probabilité adéquate. Supposons par exemple qu'à une étape donnée on doive choisir une réécriture du symbole T pour aboutir à un mot de longueur j. Si $T \to T' \mid T''$, on choisit d'engendrer un mot de longueur j dérivant de T' avec probabilité $T'(j)/T(j)$, ou dérivant de T'' avec probabilité $T'(j)/T(j)$. Si $T \to T'T''$, on choisit un entier h avec probabilité $T'(h)T''(j-h)/T(h)$, puis on engendre un mot dérivant de T' de longueur h concaténé à un mot dérivant de T'' de longueur $j-h$. On trouvera les détails du processus dans [10].

Les meilleurs algorithmes dérivés de cette méthode effectuent une génération d'un mot de longueur n en $O(n)$ opérations arithmétiques après un prétraitement en $O(n^2)$ [19], ou bien un génération d'un mot en $O(n \log n)$ opérations arithmétiques après un prétraitement en $O(n)$ [11]. Notons cependant que les nombres manipulés sont extrêmement grands ; par conséquent, il convient de s'intéresser à la complexité machine (« *bit complexity* »). On peut trouver dans [4] une discussion à ce propos.

Cette méthode s'applique bien évidemment aux langages rationnels. Toutefois, dans ce cas particulier, une approche légèrement différente permet d'effectuer une génération en $O(n)$

opérations arithmétiques avec un prétraitement linéaire aussi [13]. Pour ce faire, on considère une grammaire régulière du langage, dans laquelle toute règle est de la forme $T \to T_1 \mid \ldots \mid T_m$, où $T_i = xT_j$ ou bien $T_i = x$ ($x \in X$), ou encore $T_i = \varepsilon$. On calcule les coefficients $T(n)$ à l'aide de la relation de récurrence suivante :

$$T \to T_1 \mid \ldots \mid T_m \quad \Rightarrow \quad T(n) = T_1(n) + \ldots + T_m(n) \tag{5}$$

où

$$\begin{aligned} T_i = xT_j \quad &\Rightarrow \quad T_i(n) = T_j(n-1) && (6)\\ T_i = x \quad &\Rightarrow \quad T_i(n) = 1 \text{ si } n = 1 \text{ ; } T_i(n) = 0 \text{ sinon.} && (7)\\ T_i = \varepsilon \quad &\Rightarrow \quad T_i(n) = 1 \text{ si } n = 0 \text{ ; } T_i(n) = 0 \text{ sinon.} && (8) \end{aligned}$$

La phase de génération s'effectue de manière similaire au cas général. Ici, chaque étape consiste simplement à choisir une lettre avec une probabilité adéquate. Cela revient à construire un chemin de longueur n dans un automate fini déterministe du langage.

3 Génération en fréquences exactes

Soit $(n_1, n_2, \ldots, n_k)$ un vecteur d'entiers tel que $n_1 + n_2 + \cdots + n_k = n$. Notre but est d'engendrer aléatoirement et uniformément un mot de L_n comportant exactement n_i lettres x_i, pour tout $1 \leq i \leq k$.

Le principe de la méthode que nous décrivons est une extension naturelle du schéma général vu dans la section précédente. Il est d'ailleurs utilisé dès qu'il s'agit d'engendrer des structures aléatoires en fixant plus d'un paramètre, par exemple en [7]. Notre but est simplement de formaliser la méthode dans le cadre de la génération de mots de langages algébriques à nombres de lettres fixés.

Cette fois encore, on suppose la grammaire sous forme normale de Chomsky. Pour tout symbole non terminal T, on note $T(j_1, j_2, \ldots, j_k)$ le nombre de mots qui dérivent de T et qui ont j_1 lettres x_1, j_2 lettres x_2, ..., j_k lettres x_k. La phase de prétraitement consiste à créer une table des $T(j_1, j_2, \ldots, j_k)$ pour $0 \leq j_1 \leq n_1$, ..., $0 \leq j_k \leq n_k$, à partir des récurrences suivantes :

$$\begin{aligned} T \to \varepsilon \quad &\Rightarrow \quad T(0, \ldots, 0) = 1 \text{ ;} && (9)\\ T \to x_i \quad &\Rightarrow \quad T(0, \ldots, 0, 1, 0, \ldots, 0) = 1 \text{ ;} && (10)\\ T \to T' \mid T'' \quad &\Rightarrow \quad T(n_1, \ldots, n_k) = T'(n_1, \ldots, n_k) + T''(n_1, \ldots, n_k) \text{ ;} && (11)\\ T \to T'T'' \quad &\Rightarrow \quad T(n_1, n_2, \ldots, n_k) = \sum_{\substack{n'_1 + n''_1 = n_1 \\ n'_2 + n''_2 = n_2 \\ \cdots \\ n'_k + n''_k = n_k}} T(n'_1, n'_2, \ldots, n'_k) U(n''_1, n''_2, \ldots, n''_k). && (12) \end{aligned}$$

Comme précédemment, on choisit à chaque étape une réécriture du symbole T courant. Cette fois il faut aboutir à un mot de distribution $\mathbf{j} = (j_1, \ldots, j_k)$. Si $T \to T' \mid T''$, on choisit d'engendrer un mot de distribution $\mathbf{j}$ dérivant de T' avec probabilité $T'(\mathbf{j})/T(\mathbf{j})$, ou dérivant de T'' avec probabilité $T''(\mathbf{j})/T(\mathbf{j})$. Si $T \to T'T''$, on choisit un vecteur $\mathbf{h} = (h_1, \ldots, h_k)$ avec probabilité $T'(\mathbf{h})T''(\mathbf{j} - \mathbf{h})/T(\mathbf{h})$, puis on engendre un mot dérivant de T' de distribution h concaténé à un mot dérivant de T'' de distribution $\mathbf{j} - \mathbf{h}$.

Le traitement du cas rationnel est similaire :

$$T \to T_1 \mid \ldots \mid T_m \quad \Rightarrow \quad T(n_1, \ldots, n_k) = T_1(n_1, \ldots, n_k) + \ldots + T_m(n_1, \ldots, n_k)$$

où

$$\begin{aligned} T_i = x_m T_j &\Rightarrow T_i(n_1,\ldots,n_m,\ldots,,n_k) = T_j(n_1,\ldots,n_m-1,\ldots,n_k)\ ; \\ T_i = x_m &\Rightarrow T_i(0,\ldots,0,1,0,\ldots,0) = 1 \text{ si } n_m = 1\ ; \\ T_i = \varepsilon &\Rightarrow T_i(0,\ldots,0) = 1. \end{aligned}$$

La pierre d'achoppement de cette méthode réside dans sa forte complexité en temps et en mémoire. Elle nécessite en effet le calcul d'une table de $\Theta(n_1 n_2 \ldots n_k)$ valeurs. De plus, dans le cas non rationnel, les convolutions multiples de la formule (12) impliquent une phase de génération particulièrement lente aussi. En revanche, si le langage est rationnel, la génération s'effectue en complexité arithmétique linéaire (mais le problème du calcul de la table reste entier).

La méthode est donc envisageable pour un alphabet de taille limitée. Elle peut aussi être étendue à alphabet de taille quelconque, à condition de limiter le nombre de contraintes. Par exemple, au lieu de fixer $(n_1, n_2, \ldots, n_k)$, on peut fixer uniquement $(n_1, \ldots, n_j)$ avec $j < k$ (les nombres d'occurrences des (k_j) autres lettres restant «libres»).

4 Génération en fréquences moyennes

Le problème que nous considérons dans cette section est différent. Il s'agit de faire en sorte que tout mot w de L_n ait une probabilité non nulle $p(w)$ d'être engendré, mais que la distribution moyenne des lettres des mots engendrés respecte le vecteur $\mathbf{v}$. En d'autres termes, il faut que

$$p(w) > 0 \quad \forall w \in L_n \tag{13}$$

et

$$\frac{1}{n} \sum_{w \in L_n} |w|_{x_i} p(w) \ \sim\ v_i \quad \forall i \in \{1, 2, \ldots, k\}. \tag{14}$$

De plus, nous imposons que deux mots ayant la même distribution de lettres aient la même probabilité d'apparition :

$$(|w|_{x_i} = |w'|_{x_i} \ \forall i \in \{1, 2, \ldots, k\}) \ \Rightarrow\ p(w) = p(w'). \tag{15}$$

La démarche que nous proposons consiste à pondérer chaque lettre de l'alphabet en définissant une application $\pi : X \to I\!R_+^*$, que nous appelons *pondération*. Elle s'étend naturellement à tous les mots de X^* :

$$\pi(w) = \prod_{1 \le i \le k} \pi(x_i)^{|w|_{x_i}},$$

ainsi qu'à tout langage fini, en particulier à L_n :

$$\pi(L_n) = \sum_{w \in L_n} \pi(w).$$

Il est clair que si l'algorithme de génération est tel que

$$p(w) = \frac{\pi(w)}{\pi(L_n)} \quad \forall w \in L_n, \tag{16}$$

alors le nombre moyen d'occurrences d'une lettre donnée dans les mots engendrés est d'autant plus grand que la pondération de cette lettre est élevée par rapport aux autres. Il est clair aussi que les conditions (13) et (15) sont respectées.

Notre problème peut maintenant être décomposé en deux phases :

1. trouver une pondération π qui satisfasse la condition (14) ;

2. cette pondération étant donnée, trouver un algorithme de génération qui respecte la condition (16).

Commençons par nous intéresser à la seconde phase. Etant donnée une pondération π, la démarche pour engendrer des mots aléatoires est identique à celle utilisée pour la génération uniforme, à ceci près que, lors de la phase de dénombrement, le poids des lettres est pris en compte. Ainsi, la règle (2) est remplacée par

$$T \to x_i \quad \Rightarrow \quad T(1) = \pi(x_i).$$

et, dans le cas d'un langage rationnel, on remplace respectivement les règles (6) et (7) par

$$\begin{aligned} T_i = xT_j \quad &\Rightarrow \quad T_i(n) = \pi(x)T_j(n-1) \\ T_i = x \quad &\Rightarrow \quad T_i(n) = \pi(x) \text{ si } n = 1 \ ; \ T_i(n) = 0 \text{ sinon.} \end{aligned}$$

L'étape de génération est identique au cas uniforme. On garantit de cette façon que la probabilité d'apparition de chaque mot sera proportionnelle à son poisd $\pi(w)$.

Remarquons que la démarche ainsi décrite est tout à fait similaire à celle consistant à utiliser une grammaire *ambiguë* pour engendrer des mots selon le schéma uniforme de la section 2 : la probabilité de chaque mot est proportionnelle au nombre de dérivations qui y mènent. Dans le cas présent, le poids $\pi(w)$ de chaque mot, bien que non nécessairement entier, représente en quelque sorte ce nombre de dérivations.

Il s'agit maintenant de trouver une pondération π qui satisfasse la condition (14). Considérons la série génératrice suivante :

$$L_\pi(t, \mathbf{x}) = \sum_{w \in L} \pi(w) t^{|w|} x_1^{|w|_{x_1}} \ldots x_k^{|w|_{x_k}},$$

où $\mathbf{x} = (x_1, x_2, \ldots, x_k)$.

Pour toute lettre $i \in \{1, \ldots, k\}$, le nombre moyen d'occurrences de la lettre x_i dans les mots engendrés par l'algorithme est égal à $n\mu_i(\pi)$, où

$$\mu_i(\pi) \quad = \quad \frac{\sum_{w \in L_n} |w|_{x_i} \pi(w)}{n\pi(L_n)} \quad = \quad \frac{[t^n]\Gamma_{\pi,x_i}(t)}{[t^n]\Gamma_\pi(t)} \tag{17}$$

avec

$$\Gamma_{\pi,x_i}(t) = \frac{\partial L_\pi(t,\mathbf{x})}{\partial x_i}(t, \mathbf{1}) \qquad \text{et} \qquad \Gamma_\pi(t) = t\frac{\partial L_\pi(t,\mathbf{x})}{\partial t}(t, \mathbf{1})$$

Notre problème consiste à trouver une pondération π telle que $\mu_i(\pi) = v_i$ pour tout $i \in \{1, \ldots, k\}$. Nous différencions deux cas, selon que cette série génératrice est algébrique (sous-section 4.1) ou rationnelle (sous-section 4.2).

4.1 Le cas algébrique général

Le résultat qui suit se déduit immédiatement d'un théorème plus général de Drmota [5, Théorème 1, p.107].

Théorème 1 *Soit* $\mathbf{y} = \mathbf{F}(t, \mathbf{y}, \mathbf{x})$ *un ensemble d'équations obtenu à partir d'une grammaire algébrique pondérée par* π*, où* $\mathbf{y} = (y_1, y_2, \ldots, y_N)$ *est le vecteur des séries génératrices associées aux symboles non terminaux et* $\mathbf{F}(t, \mathbf{y}, \mathbf{x}) = (F_1(t, \mathbf{y}, \mathbf{x}), \ldots, F_N(t, \mathbf{y}, \mathbf{x})$*. Supposons que*

- $\mathbf{F}(0, \mathbf{y}, \mathbf{x}) = 0$ *et* $\mathbf{F}(t, \mathbf{0}, \mathbf{x}) \neq 0$ *;*

- *le système d'équations ne peut pas se ramener à un système d'équations linéaires en* $\mathbf{y}$ *;*
- *le système d'équations est* simple *; c'est-à-dire qu'il existe un ensemble de N cônes $(k+1)$-dimensionnels $C_i \subseteq \mathbb{R}^{k+1}$ tels que, pour tout $(n, m_1, \ldots, m_k) \in C_j$ tel que n, m_1, $\ldots$, m_k soient suffisamment grands, le coefficient du terme $t^n x_1^{m_1} \ldots x_k^{m_k}$ soit non nul;*
- *la grammaire est* fortement connexe *; c'est-à-dire que tout symbole non terminal peut être obtenu par dérivations à partir de tout autre symbole non terminal.*

S'il existe une solution positive $(\mathbf{y}_0, t_0)$ au système d'équations

$$\left\{ \begin{array}{lcl} \mathbf{y} & = & \mathbf{F}(t, \mathbf{y}, \mathbf{1}) \\ 0 & = & \det(I - \mathbf{F}_{\mathbf{y}}(t, \mathbf{y}, \mathbf{1}) \end{array} \right.$$

dans lequel $\mathbf{F}_{\mathbf{y}}$ représente la matrice $\left(\frac{\partial F_i}{\partial y_j}\right)_{1 \le i,j \le N}$, alors

$$\mu_i(\pi) = \frac{1}{t(1)} \frac{\partial t}{\partial x_i}(\mathbf{1})$$

pour tout $1 \le i \le k$, où $t(\mathbf{x})$ est une solution du système d'équations

$$\left\{ \begin{array}{lcl} \mathbf{y} & = & \mathbf{F}(t, \mathbf{y}, \mathbf{x}) \\ 0 & = & det(I - \mathbf{F}_{\mathbf{y}}(t, \mathbf{y}, \mathbf{x}) \end{array} \right. \tag{18}$$

telle que $t(\mathbf{1}) = t_0$.

Ce théorème est applicable à notre problème, en considérant par exemple que la fonction génératrice $L_\pi(t, \mathbf{x})$ est le premier élément du vecteur $\mathbf{y}$. Remarquons toutefois que le résultat ne s'applique pas si la série génératrice de notre langage est rationnelle, car dans ce cas le système d'équations se ramène à un système linéaire.

Voici un exemple d'application. Il s'agit d'engendrer des arbres plans (arbres enracinés dans lesquels les fils d'un même père sont ordonnés) à n arêtes, en fixant le nombre moyen de feuilles à μn $(0 < \mu < 1)$. Une bijection classique permet de ramener le problème à la génération de mots de longueur $2n$ du langage défini par la grammaire $S \to aSbS + aSb + cdS + cd$, en fixant le nombre moyen de lettres c. Munissons donc cette lettre c d'un poids π_1, réel positif: $S \to aSbS + aSb + \pi_1 cdS + \pi_1 cd$. La grammaire ainsi pondérée satisfait aux conditions du théorème. On obtient (en ne considérant qu'une lettre sur deux car elles vont par paires)

$$\left\{ \begin{array}{lcl} F(t, y, x_1) & = & xy^2 + xy + \pi_1 z_1 xy + \pi_1 z_1 x \\ F_y(t, y, x_1) & = & 2\,xy + x + \pi_1 z_1 x. \end{array} \right.$$

La résolution du système (18) donne

$$\pi_1 = \left(\frac{\mu}{1-\mu}\right)^2 \quad \text{avec } 0 < \mu < 1.$$

Cela signifie que, pour engendrer un mot à $2n$ lettres a (resp. un arbre à n arêtes) comportant μn lettres c (resp. feuilles) en moyenne, il faut prendre $\pi_1 = (\mu/(1-\mu))^2$. Nous devons toutefois reconnaître que cet exemple est particulièrement simple. Dans le cas général, il est rare que nous sachions résoudre le système (18).

4.2 Le cas rationnel

Si L est un langage rationnnel, les fonctions $\Gamma_{\pi,x_i}(t)$ et $\Gamma_\pi(t)$ de l'expression (17) sont des fractions rationnelles. La valeur du $n$ième coefficient du développement en série de chacune d'entre elles dépend donc des zéros de son dénominateur (voir par exemple [12, p.360]). Si l'on se contente d'une équivalence asymptotique de cette valeur, il suffit de connaître le zéro réel de plus petit module s de ce dénominateur et sa multiplicité ν. Cependant, dans notre cas, le dénominateur dépend des $\pi(x_i)$ qui sont nos inconnues. *A priori*, on se retrouve donc confronté au même problème que dans le cas algébrique : résoudre une équation algébrique en s dont les coefficients sont paramétrés par la pondération π inconnue. Toutefois, la proposition suivante, dont la preuve est aisée, va nous tirer d'affaire.

Proposition 2 *Soit L un langage rationnel sur un alphabet $X = \{x_1, x_2, \ldots, x_k\}$. Soient π une pondération de X, s l'unique pole réel de plus petit module de $\Gamma_\pi(t)$ (resp. de $\Gamma_{\pi,x_i}(t)$), et ν sa multiplicité . Alors $\mu_i(s\pi) = \mu_i(\pi)$ pour tout $i = 1, \ldots, k$, et l'unique pole de plus petit module de $\Gamma_{s\pi}(t)$ (resp. de $\Gamma_{s\pi,x_i}(t)$) est 1, de multiplicité ν.*

Ceci signifie que, s'il existe une pondération satisfaisant nos critères de fréquence moyenne et si le pôle réel de plus petit module s de $\Gamma_\pi(t)$ est égal à celui de $\Gamma_{\pi,x_i}(t)$, alors il existe une pondération telle que ce pôle réel est égal à 1. Cela nous autorise à poser *a priori* $s = 1$.

Il reste maintenant à déterminer dans quelles conditions ce pôle s est commun à $\Gamma_\pi(t)$ et à tous les $\Gamma_{\pi,x_i}(t)$. C'est l'objet du théorème suivant.

Théorème 3 *Soit L un langage rationnel sur un alphabet $X = \{x_1, \cdots, x_k\}$, π une pondération de X et $\mathcal{A}$ l'automate fini déterministe minimal reconnaissant L. Soit $C = \{C_1, \cdots, C_f\}$ l'ensemble des composantes fortement connexes maximales de cet automate. Soit enfin s le pôle dominant, supposé unique, de $L_\pi(t, 1)$. Si chaque C_j $(1 \leq j \leq f)$ non réduite à un seul sommet contient toutes les lettres de X, alors la série $\Gamma_\pi(t)$ et toutes les séries $\Gamma_{\pi,x_i}(t)$ $(1 \leq i \leq k)$ ont s pour pôle dominant avec multiplicité $\nu + 1$, où ν est la multiplicité du pôle dominant de la série génératrice non pondérée du langage L.*

Ce théorème est une conséquence du théorème de Perron-Frobenius. Il se prouve de façon similaire au théorème 2 de [21].

Ces considérations nous permettent de présenter un algorithme de recherche d'une pondération π qui permette de satisfaire la condition (14).

1. Calculer la série $L_\pi(t, \mathbf{x})$ (dans laquelle les $\pi(x_i)$ sont des paramètres dont les valeurs sont inconnues). En déduire $\Gamma_\pi(t)$ et $\Gamma_{\pi,x_i}(t)$ pour tout $i = 1, \ldots, k$. Notons s le pôle dominant de ces séries.

2. Poser $s = 1$ et prendre ν comme dans le théorème qui précède. Calculer avec ces valeurs un équivalent asymptotique $\bar{\mu}_i(\pi)$de chacun des $\mu_i(\pi)$ selon la formule (17).

3. Résoudre en $(\pi(x_1), \ldots, \pi(x_k))$ le système algébrique suivant:

$$\left\{\begin{array}{lcl} \bar{\mu}_1(\pi) & = & v_1 \\ & \vdots & \\ \bar{\mu}_{k-1}(\pi) & = & v_{k-1} \\ D_\pi(1, \mathbf{1}) & = & 0 \end{array}\right.$$

   où $D_\pi(t, \mathbf{x})$ désigne le dénominateur de la série $L_\pi(t, \mathbf{x})$. (La dernière équation traduit le fait que 1 est un pôle des séries Γ.)

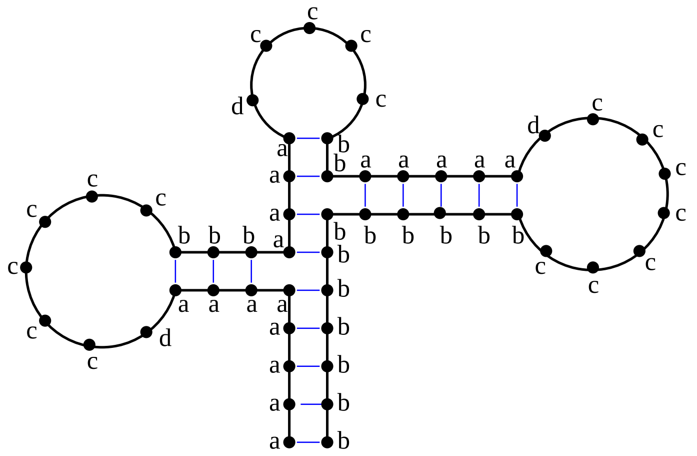

FIG. 1 – *Représentation de la structure secondaire qui correspond au mot* aaaaaaaadcccccbbb-aaaadccccbbaaaaadcccccccbbbbbbbbbbb.

4. Parmi les solutions de ce système, en chercher une telle que
   - tous les $\pi(x_i)$ soient strictement positifs ;
   - le pôle 1 soit le pôle réel de plus petit module de $L_\pi(t, \mathbf{1})$. (Nous avons imposé qu'il soit un pôle, mais pas qu'il soit le pôle dominant.)

Remarquons que le système à résoudre ne dépend d'aucun paramètre. il peut donc être résolu numériquement, par exemple avec un outil tel que GB [8]. Si aucune solution satisfaisant le point 4 n'est trouvée, c'est qu'il n'existe pas de pondération satisfaisant la condition (17).

5 Exemples en génomique

Notre travail offre la possibilité d'engendrer des séquences obéissant à la fois à un modèle de Bernoulli et à une syntaxe. Il existe par ailleurs des possibilités d'engendrer des séquences aléatoires selon des modèles syntaxiques. En effet, il n'est pas difficile d'utiliser une HMM ou une grammaire stochastique [6] pour engendrer des séquences. Par exemple, le programme *HmmerEmit* de la boîte à outils d'analyse des génomes *GCG* construit des séquences qui respectent un profil HMM donné. L'apport majeur des grammaires pondérées est d'offrir une expressivité et une précision plus larges que ces modèles classiques.

Voici un exemple simple de l'usage que l'on peut faire d'une grammaire pondérée. Il s'agit de modéliser des structures secondaires simplifiées, composées seulement de tiges (sans mésappariement) terminées par des boucles comportant au moins cinq nucléotides. Nous choisissons de représenter chaque structure par un mot : les nucléotides en $5'$ et $3'$ d'un appariement sont représentés respectivement par les lettres a et b (qui peuvent être considérées comme des parenthèses ouvrante et fermante) ; chaque nucléotide à l'intérieur d'une boucle est représenté par la lettre c, sauf le premier à qui on attribue la lettre d. La figure 1 présente un mot et la structure qui lui correspond. Cette modélisation est une variante de la représentation des structures secondaires par des mots du langage de Motzkin [30, 31, 33]. Le langage des structures secondaires ainsi définies est algébrique ; en voici une grammaire :

$$\begin{array}{rcl} S & \rightarrow & \mathtt{a}\,S\,\mathtt{b}\,S \mid \mathtt{a}\,S\,\mathtt{b} \mid \mathtt{d}\,T\,\mathtt{c} \\ T & \rightarrow & \mathtt{c}\,T \mid \mathtt{c\,c\,c} \end{array}$$

Le rôle de la lettre d est de permettre de faire varier le nombre moyen de tiges-boucles des structures aléatoires en jouant sur la pondération. De même, la longueur moyenne des boucles peut être modulée en modifiant le poids de la lettre c. Supposons par exemple que l'on veuille engendrer des structures de longueur 60 nucléotides. En calculant la série génératrice et appliquant la formule de calcul des $\mu_i(\pi)$ de la page 6, on montre aisément que, si tous les poids valent 1 (c'est-à-dire si la génération est uniforme), alors le nombre moyen de boucles est environ égal à 5, 62, et chaque boucle comporte 6, 68 nucléotides en moyenne. En posant $\pi(\mathtt{a}) = 16$, $\pi(\mathtt{c}) = 2$ et $\pi(\mathtt{b}) = \pi(\mathtt{d}) = 1$, on obtient des structures ayant en moyenne trois boucles de longueur 5, 80 nucléotides. Si l'on veut compter *exactement* trois boucles dans chaque structure engendrée, on peut concevoir une grammaire différente :

$$
\begin{array}{lcl}
S & \rightarrow & \mathtt{a}\, S\, \mathtt{b} \mid T\, T\, T \mid T\, U \mid U\, T \\
T & \rightarrow & \mathtt{a}\, T\, \mathtt{b} \mid \mathtt{a}\, V\, \mathtt{b} \\
U & \rightarrow & \mathtt{a}\, U\, \mathtt{b} \mid \mathtt{a}\, T\, T\, \mathtt{b} \\
V & \rightarrow & \mathtt{c}\, V \mid \mathtt{c\, c\, c\, c\, c}
\end{array}
$$

Ici, il est inutile d'ajouter une lettre d : le nombre de boucles est fixé par la grammaire. Le poids de c détermine la longueur moyenne des boucles. Ces grammaires peuvent être complexifiées pour modéliser des structures de façon plus «réaliste». Ainsi, par exemple, on peut modéliser des mésappariements dans les tiges en ajoutant la règle $T \rightarrow \mathtt{x}\, T\, \mathtt{x}$ à la grammaire ci-dessus.

Un logiciel de génération de séquences, baptisé GenRGenS (Generation of Random Genomic Sequences) est en cours de développement [23, 28]. Notre projet est d'en faire un outil capable de prendre en compte aussi bien les modèles simples que les modèles hétérogènes les plus complexes. La version actuelle de GenRGenS prend en compte le modèle markovien (avec possibilité de tenir compte d'une phase quelconque), certaines chaînes de Markov cachées et les grammaires pondérées. Les modèles sont décrits par des fichiers. Voici par exemple un fichier de description pour la première grammaire pondérée définie ci-dessus.

```
TYPE = GRAMMAR
SYMBOLS = LETTERS
RULES
  S -> a S b S;
  S -> a S b;
  S -> d T c;
  T -> c T;
  T -> c c c;
WEIGHTS =
  a 16
  c 2
```

Le fichier suivant décrit un modèle markovien d'ordre un à trois phases. Les nombres qui suivent les dinucléotides représentent les fréquences observées sur la séquence de référence dans les phases 0, 1 et 2.

```
TYPE = MARKOV 1 3
SYMBOLS = LETTERS
FREQUENCIES =
  aa 2 1 0   ac 3 1 3   ag  5 7 5   at 5 4 4
  ca 4 3 2   cc 2 3 5   cg  1 0 0   ct 0 1 1
  ga 5 5 6   gc 8 6 7   gg  2 8 9   gt 1 2 3
  ta 2 5 7   tc 3 6 7   tg  4 1 0   tt 2 1 1
```

Les chaînes de Markov cachées peuvent être gérées grâce à un mécanisme d'*alias* qui permet de définir une chaîne sur un alphabet étendu, puis de substituer certaines lettres à d'autres en fin de traitement. Une particularité importante (en cours d'implémentation) du programme est la possibilité d'engendrer des séquences composites, c'est-à-dire dont les différentes zones obéissent à des modèles de natures très différentes. Voici par exemple un modèle grammatical très fruste d'un chromosome d'eucaryote inférieur.

```
TYPE = GRAMMAR
SYMBOLS = WORDS
RULES
  Chromosome -> Telomere5' Reste Centromere Reste Telomere3';
  Reste -> Intergenique ZonePromotrice Gene ZoneTerminatrice Reste;
  Reste -> ;
  Gene -> GeneProteique;
  Gene -> tRNA;
  Gene -> rRNA;
  Gene -> snRNA;
```

Chaque symbole terminal (`Telomere5'`, `Centromere`, `Intergenique`, `ZonePromotrice`, `GeneProteique`, etc.) de cette grammaire représente un type de zone. On peut s'arrêter là (et engendrer des mots sur l'alphabet des symboles terminaux de cette grammaire), ou bien aller plus loin en associant à chaque type de zone un modèle particulier (par exemple markovien pour les zones intergéniques, grammatical pour les tRNA). Le résultat de la génération sera une séquence sur l'alphabet des nucléotides.

Références

[1] S.F. Altschul, T.L. Madden, A.A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D.J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, 1997.

[2] G. Benson. Tandem repeats finder: a program to analyse DNA sequences. *Nucleic Acids Research*, 27(2):573–580, 1999.

[3] E. Coward. Shufflet: Shuffling sequences while conserving the k-let counts. *Bioinformatics*, 15:1058–1059, 1999. Programme : http://www.genetique.uvsq.fr/eivind/shufflet.html.

[4] A. Denise and P. Zimmermann. Uniform random generation of decomposable structures using floating-point arithmetic. *Theoretical Computer Science*, 218:233–248, 1999.

[5] M. Drmota. Systems of functional equations. *Random Structures and Algorithms*, 10:103–124, 1997.

[6] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis. Probabilistic models of proteins and nucleic acids.* Cambridge University Press, 1998.

[7] I. Dutour and J.-M. Fédou. Object grammars and random generation. *Discrete Mathematics and Theoretical Computer Science*, 2:47–61, 1998.

[8] J-.C. Faugère. GB. http://calfor.lip6.fr/GB.html.

[9] W.M. Fitch. Random sequences. *Journal of Molecular Biology*, 163:171–176, 1983.

[10] Ph. Flajolet, P. Zimmermann, and B. Van Cutsem. A calculus for the random generation of labelled combinatorial structures. *Theoretical Computer Science*, 132:1–35, 1994.

[11] M. Goldwurm. Random generation of words in an algebraic language in linear binary space. *Information Processing Letters*, 54:229–233, 1995.

[12] R.L. Graham, D.E. Knuth, and O. Patashnik. *Mathématiques concrètes.* ITP France (Vuibert), 2ème edition, 1997.

[13] T. Hickey and J. Cohen. Uniform random generation of strings in a context-free language. *SIAM J. Comput.*, 12(4):645–655, 1983.

[14] D. Kandel, Y. Matias, R. Unger, and P. Winkler. Shuffling biological sequences. *Discrete Applied Mathematics*, 71:171–185, 1996.

[15] M. Klaerr-Blanchard, H. Chiapello, and E. Coward. Detecting localized repeats in genomic sequences: A new strategy and its application to *B. subtilis* and *A. thaliana*. *Comput. Chem.*, 24(1):57–70, 2000.

[16] D.J. Lipman and W.R. Pearson. Rapid and sensitive protein similarity searches. *Science*, 227:1435–1441, 1985.

[17] D.J. Lipman, W.J. Wilbur, T.F. Smith, and M.S. Waterman. On the statistical signifiance of nucleic acid similarities. *Nucleic Acids Research*, 12:215–226, 1984.

[18] T.M. Lowe and S.R. Eddy. tRNAscan-SE: a program for improved detection of transfer tRNA genes in genomic sequence. *Nucleic Acids Research*, 25:955–969, 1997.

[19] Harry G. Mairson. Generating words in a context free language uniformly at random. *Information Processing Letters*, 49:95–99, 1994.

[20] F. Muri-Majoube and B. Prum. Une approche statistique de l'analyse des génomes. *La Gazette des Mathématiciens*, 89:63–98, 2001.

[21] P. Nicodème, B. Salvy, and Ph. Flajolet. Motif statistics. In *European Symposium on Algorithms-ESA99*, volume 1643, pages 194–211. Lecture Notes in Computer Science, 1999.

[22] A. Nijenhuis and H.S. Wilf. *Combinatorial algorithms.* Academic Press, New York, 2nd edition, 1978.

[23] Y. Ponty. Génération aléatoire de structures génomiques : GenRGenS. Rapport de TER-stage, maîtrise d'informatique, université Paris-Sud XI, juin 2001.

[24] B. Prum, F. Rodolphe, and É. de Turckheim. Finding words with unexpected frequencies in DNA sequences. *J. Royal Statist. Soc. Ser. B*, 57:205–220, 1995.

[25] M. Régnier. A unified approach to word occurrence probabilities. *Discrete Applied Mathematics*, 104(1):259–280, 2000.

[26] G. Reinert, S. Schbath, and M.S. Waterman. Probabilistic and statistical properties of words: An overview. *Journal of Computational Biology*, 7(1–2):1–46, 2000.

[27] M.-F. Sagot. L'espoir est cette chose avec des plumes – Algorithmes combinatoires et biologie moléculaire. Habilitation à diriger les recherches, 2000.

[28] F. Sarron. GenRGenS v0.6 : notes de développement et de maintenance. Rapport de stage, université Paris-Sud XI, septembre 2000.

[29] S. Schbath. An efficient statistic to detect over- and under-represented words in DNA sequences. *Journal of Computational Biology*, pages 189–192, 1997.

[30] W. Schmitt and M.S. Waterman. Linear trees and RNA secondary structures. *Discrete Applied Mathematics*, 51:317–323, 1994.

[31] D.B. Searls. Formal language theory and biological macromolecules. *Series in Discrete Mathematics and Theoretical Computer Science*, 47:117–140, 1999.

[32] A. Vanet, L. Marsan, and M.-F. Sagot. Promoter sequences and algorithmical methods for identifying them. *Res. Microbiol.*, 150:779–799, 1999.

[33] M. Vauchaussade de Chaumont and X.G. Viennot. Enumeration of RNA's secondary structures by complexity. In V. Capasso, E. Grosso, and S.L. Paven-Fontana, editors, *Mathematics in Medecine and Biology*, volume 57 of *Lecture Notes in Biomathematics*, pages 360–365, 1985.